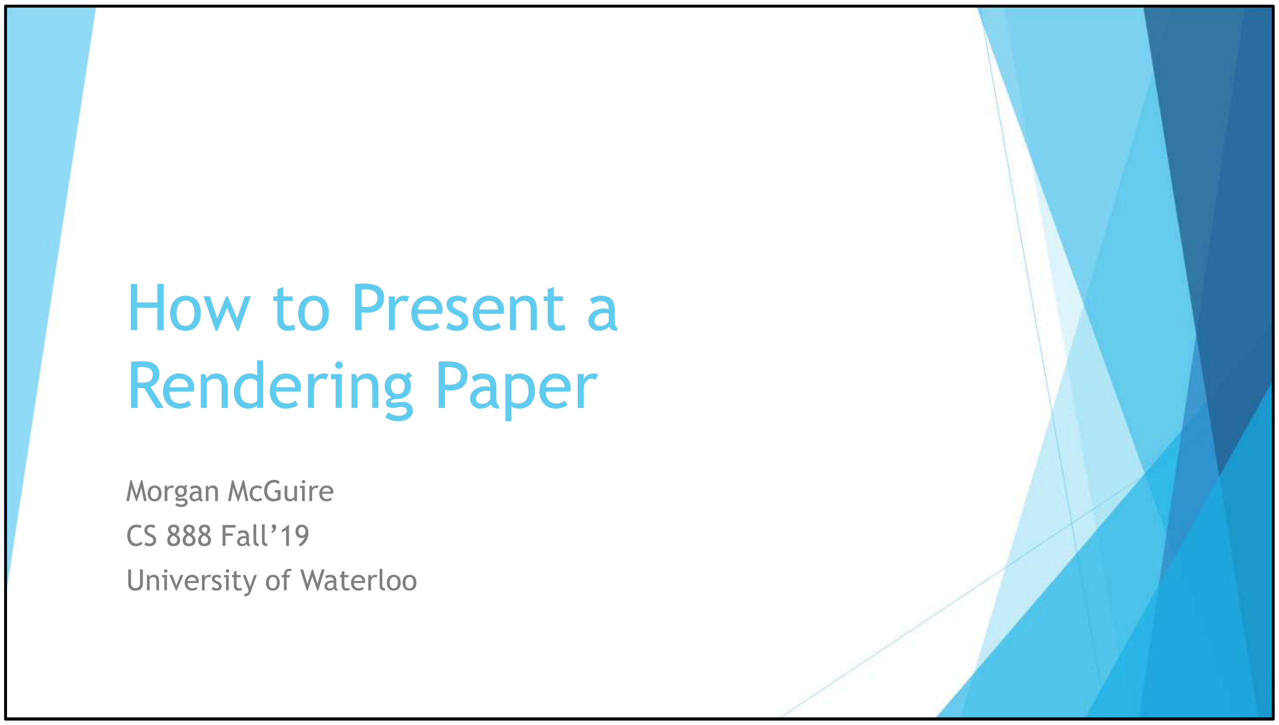
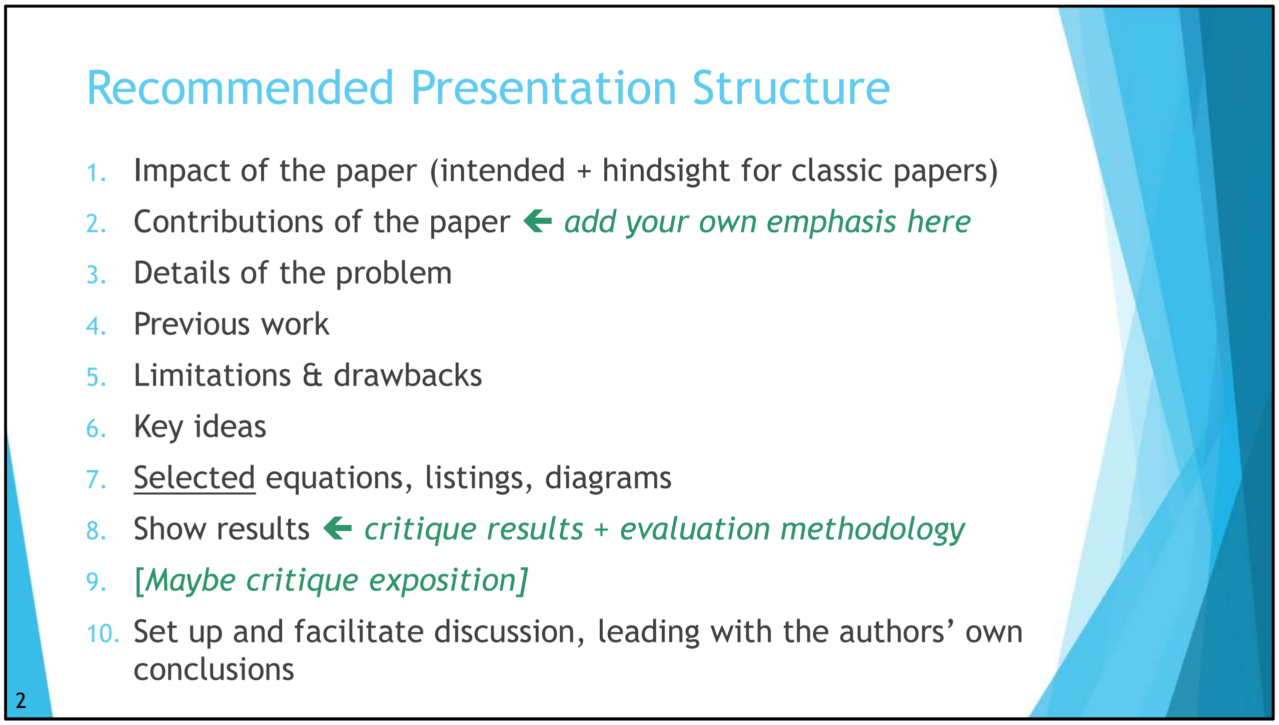


How to Present a Rendering Paper

Morgan McGuire
CS 888 Fall'19
University of Waterloo

[These are lecture notes from a graduate course, which were prepared with feedback from co-teachers Stephen Mann and Craig Kaplan and colleagues Peter Shirley and Ward Lopes]



Recommended Presentation Structure

1. Impact of the paper (intended + hindsight for classic papers)
2. Contributions of the paper ← *add your own emphasis here*
3. Details of the problem
4. Previous work
5. Limitations & drawbacks
6. Key ideas
7. Selected equations, listings, diagrams
8. Show results ← *critique results + evaluation methodology*
9. *[Maybe critique exposition]*
10. Set up and facilitate discussion, leading with the authors' own conclusions

2

Target 20 minutes of you speaking for a ~45 minute presentation, including interruptions and discussion

"I'm not sure that I understood this part" is fine

Periodically stop and invite questions

Use screen-grabs directly from the paper where possible for equations and images (to save your time as well as quote the work)

Choose what to focus on and what to mention only in passing. Do not try to present every aspect or every detail.

Respectfully critique the exposition (What made it easy to understand and should be advocated? What could have been improved, in retrospect?)

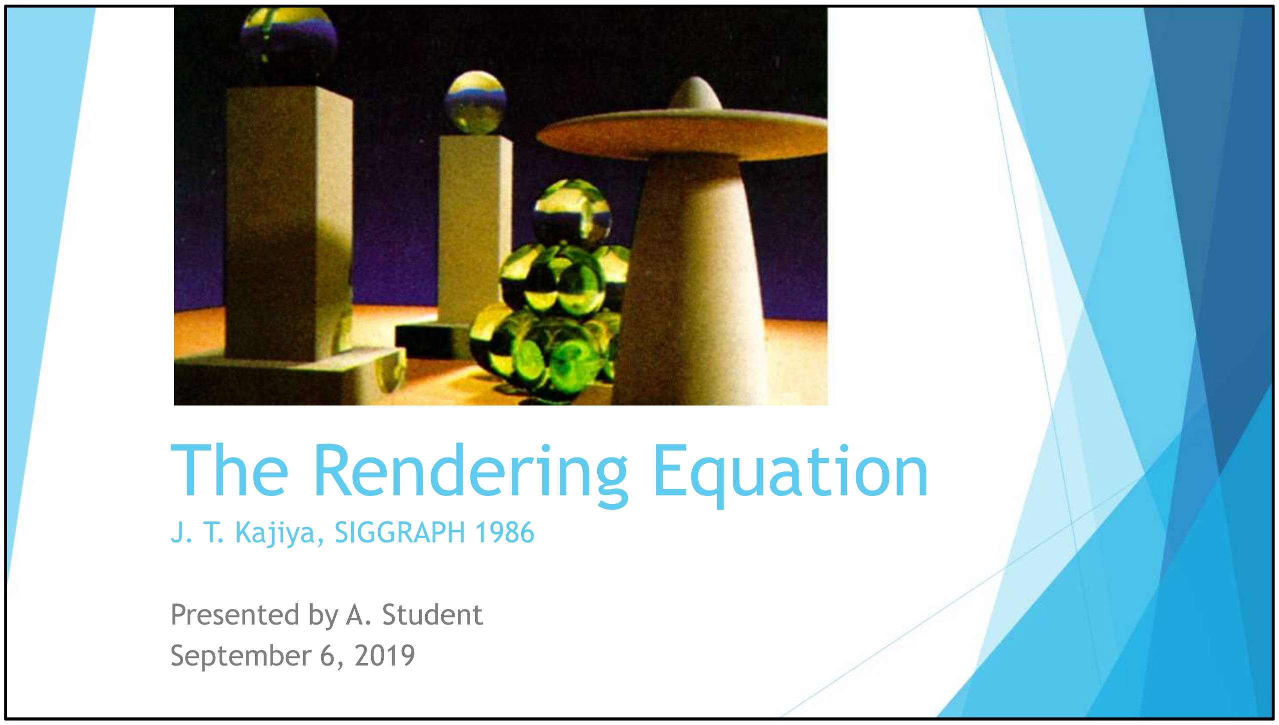
Be sure to explain the notation

Some good discussion topics:

- What you didn’t understand
- Clever, reusable tricks
- Relationship to other papers
- Potential for followup work

I’ll now give a brief example

Example



The Rendering Equation

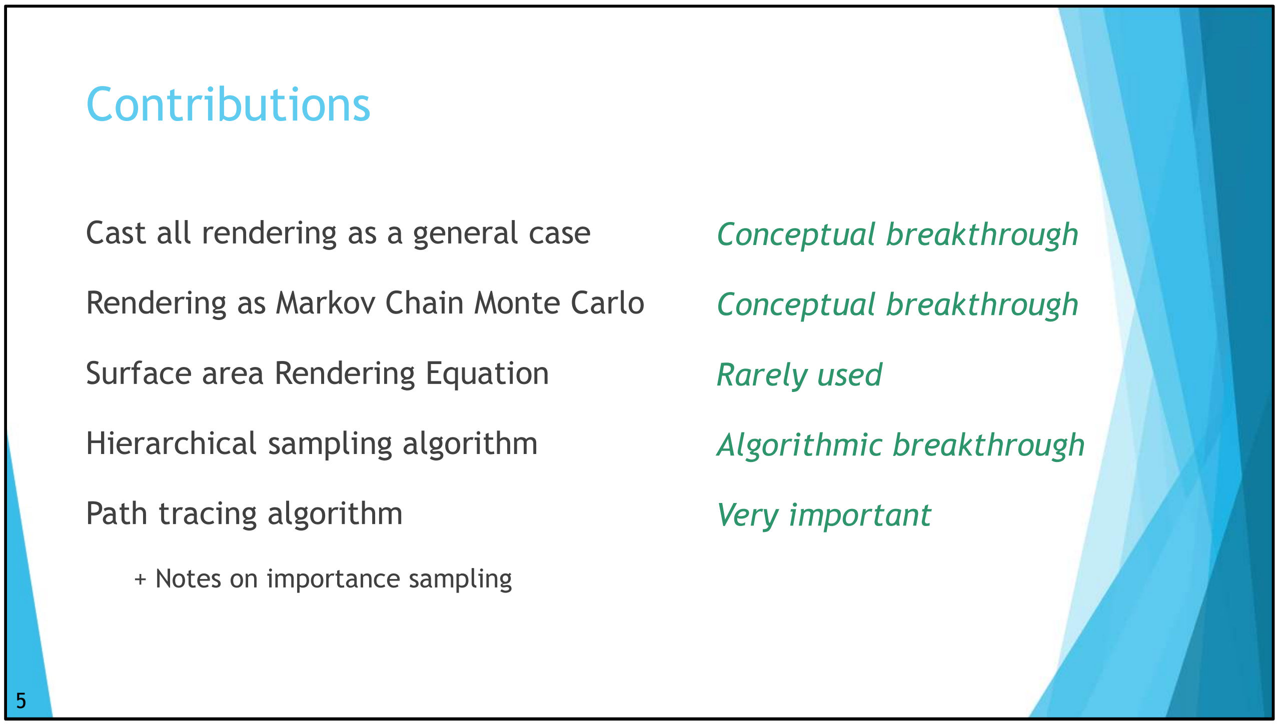
J. T. Kajiya, SIGGRAPH 1986

Presented by A. Student
September 6, 2019

The impact of this paper was enormous. It is probably the most famous paper in physically-based rendering, and underlies that whole field today. It builds on the shoulders of giants but crosses two critical thresholds to make the important breakthroughs.

The paper is written with an understanding of the significance of its core result: framing rendering as a general case for Monte Carlo integral equation solvers, rather than tackling individual phenomena separately.

It downplays the significance of the particular solution of path tracing, so the impact of that aspect comes to us from history and not the paper itself.



Contributions

Cast all rendering as a general case	*Conceptual breakthrough*
Rendering as Markov Chain Monte Carlo	*Conceptual breakthrough*
Surface area Rendering Equation	*Rarely used*
Hierarchical sampling algorithm	*Algorithmic breakthrough*
Path tracing algorithm	*Very important*
+ Notes on importance sampling	

5

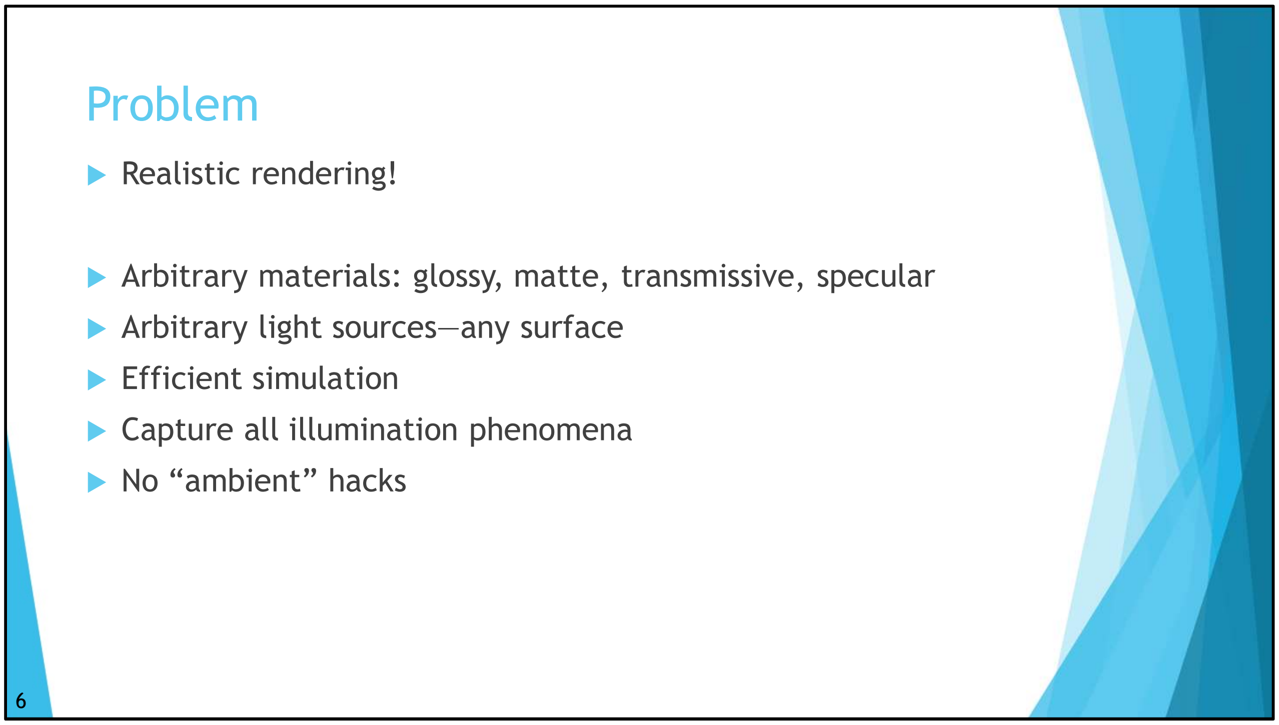
Here are the contributions as presented in the paper.

I consider the generalization of rendering and presenting it in the context of Maxwell's equations to be a conceptual breakthrough for the field. This is "the rendering equation".

The actual form of the equation as given turns out to be less useful than the concept because it has an awkward parameterization over areas.

The hierarchical sampling algorithm is a good idea, but for this application it was supplanted by quasi-Monte Carlo methods and so I won't focus on that today.

The path tracing algorithm generalizes ideas from distribution ray tracing and radiosity to create an algorithmic breakthrough, and the (unimplemented) notes about importance sampling are also very relevant.
Part of the path tracing derivation is showing that rendering is an MCMC problem. The Radiosity algorithm previously leveraged the Markov aspect, but didn't connect it to other work in monte carlo approximations.

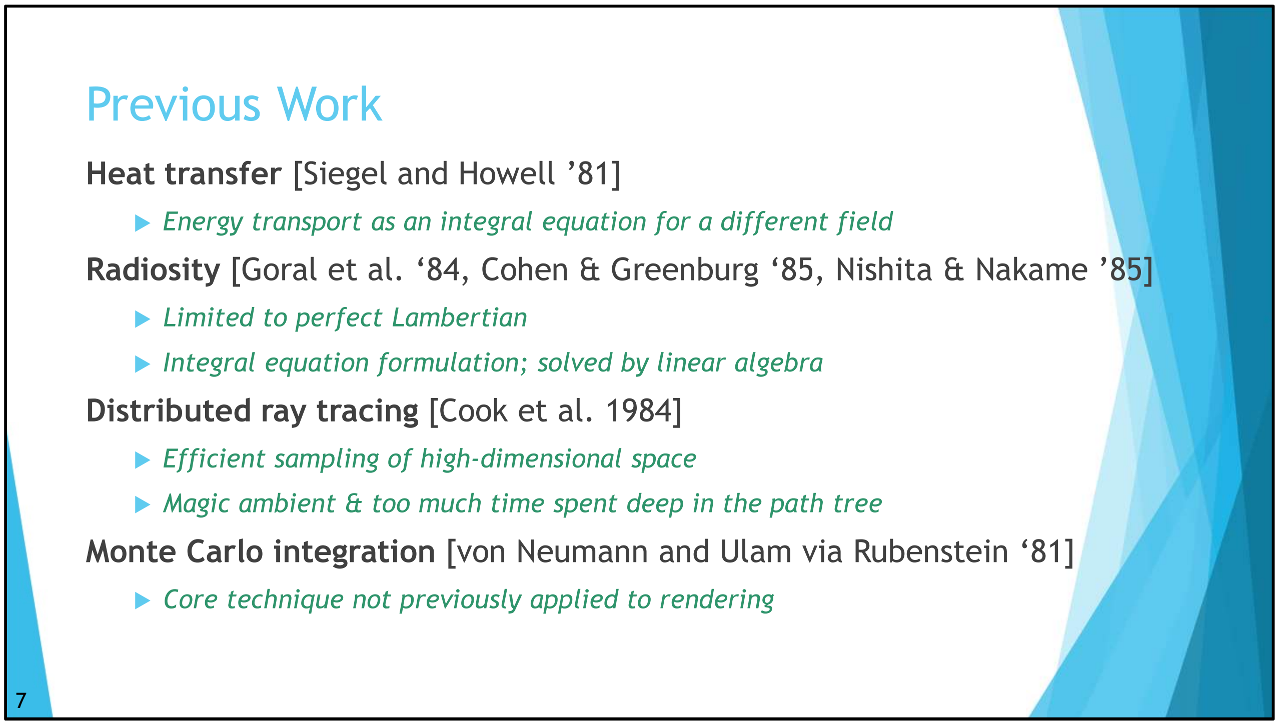


Problem

- Realistic rendering!

- Arbitrary materials: glossy, matte, transmissive, specular
- Arbitrary light sources—any surface
- Efficient simulation
- Capture all illumination phenomena
- No “ambient” hacks

6



Previous Work

Heat transfer [Siegel and Howell ’81]

- *Energy transport as an integral equation for a different field*

Radiosity [Goral et al. ‘84, Cohen & Greenburg ‘85, Nishita & Nakame ’85]

- *Limited to perfect Lambertian*
- *Integral equation formulation; solved by linear algebra*

Distributed ray tracing [Cook et al. 1984]

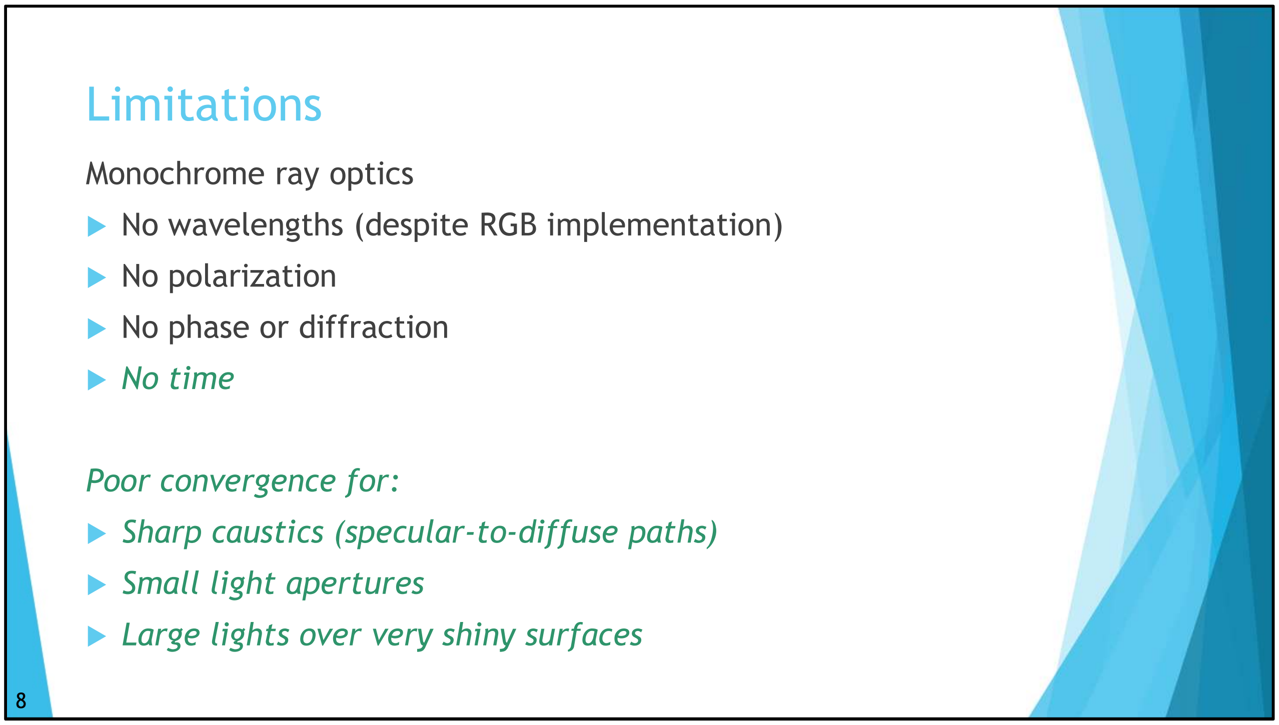
- *Efficient sampling of high-dimensional space*
- *Magic ambient & too much time spent deep in the path tree*

Monte Carlo integration [von Neumann and Ulam via Rubenstein ‘81]

- *Core technique not previously applied to rendering*

7

Questions?



Limitations

Monochrome ray optics

- No wavelengths (despite RGB implementation)
- No polarization
- No phase or diffraction
- *No time*

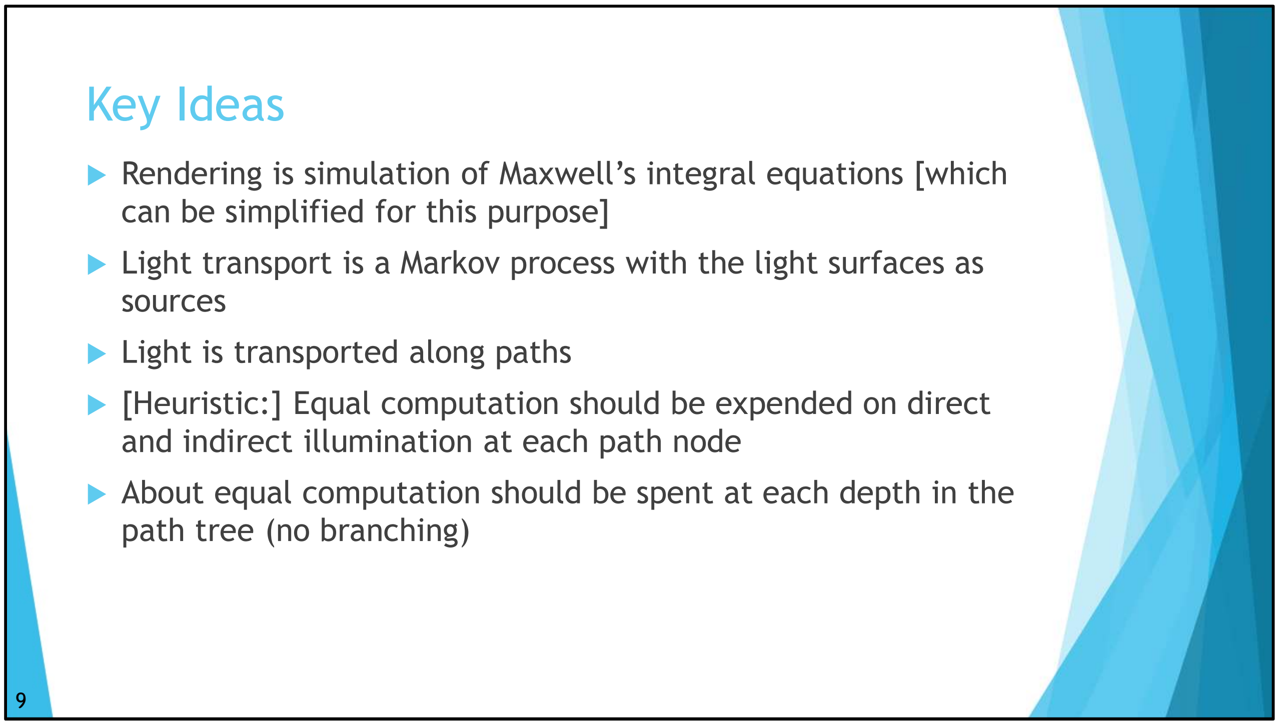
Poor convergence for:

- *Sharp caustics (specular-to-diffuse paths)*
- *Small light apertures*
- *Large lights over very shiny surfaces*

8

The paper mentions the limitation to monochrome ray optics of the theory and resulting algorithm.

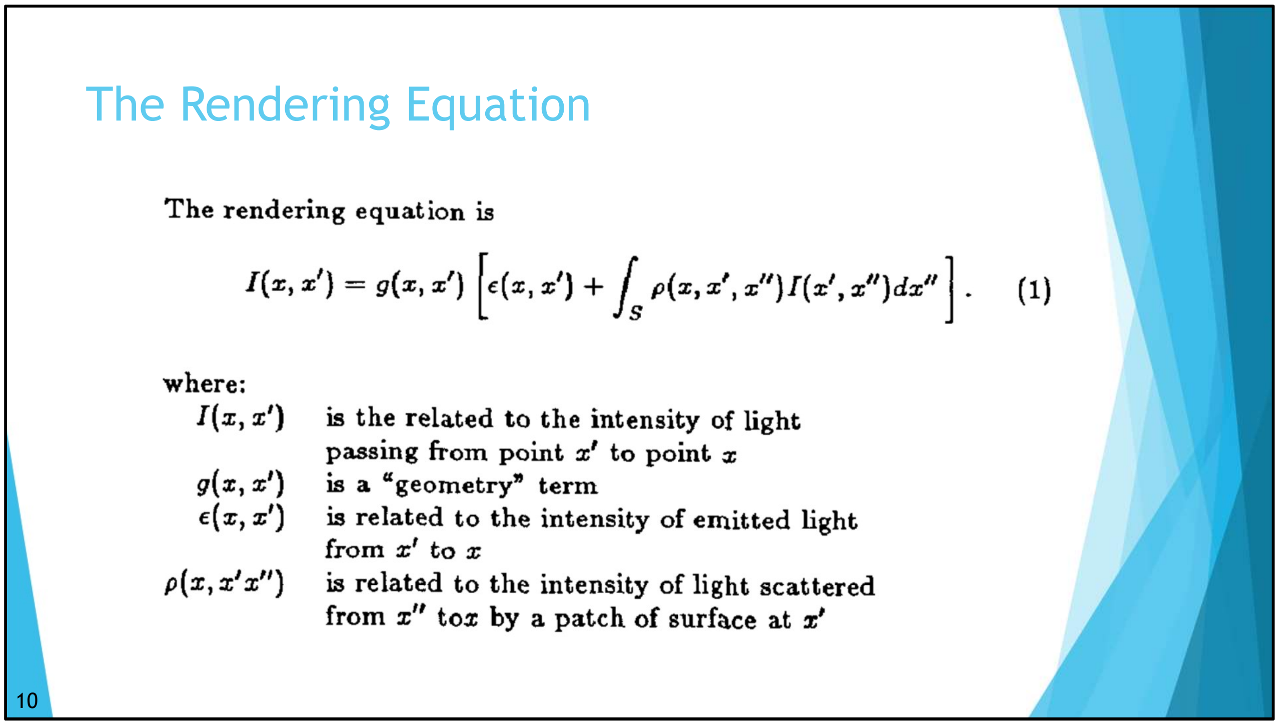
It doesn’t mention (and Kajiya likely was not aware at the time of) the poor convergence properties in scenes where the paths with the most energy transported are hard to discover by searching backwards from the camera or where importance sampling is difficult. [We’ll see later that Veach and Guibas address these limitations a decade later with an alternative algorithm.]



Key Ideas

- Rendering is simulation of Maxwell's integral equations [which can be simplified for this purpose]
- Light transport is a Markov process with the light surfaces as sources
- Light is transported along paths
- [Heuristic:] Equal computation should be expended on direct and indirect illumination at each path node
- About equal computation should be spent at each depth in the path tree (no branching)

9



The Rendering Equation

The rendering equation is

$$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]. \quad (1)$$

where:

$I(x,x')$	is the related to the intensity of light passing from point x' to point x
$g(x,x')$	is a "geometry" term
$\epsilon(x,x')$	is related to the intensity of emitted light from x' to x
$\rho(x,x'x'')$	is related to the intensity of light scattered from x'' to$x$ by a patch of surface at x'

10

Kajiya’s own words and notation explain this for you quite clearly. The key point is that this is an integral equation: I is written in terms of the integral of I.

We’re computing the light “intensity” transmitted from point x to x’, which arrived at x from the points x’’ in S.

As a guide for the modern reader,

G is the cosine of the angle of the incidence with an extra area term in the denominator, which is typically separated from emittance and put into the integrand today; he had it outside and the unusual units as an artifact of that point parameterization of the integral.

S is the set of all surface points *that are visible from x* [which will be determined by ray tracing]

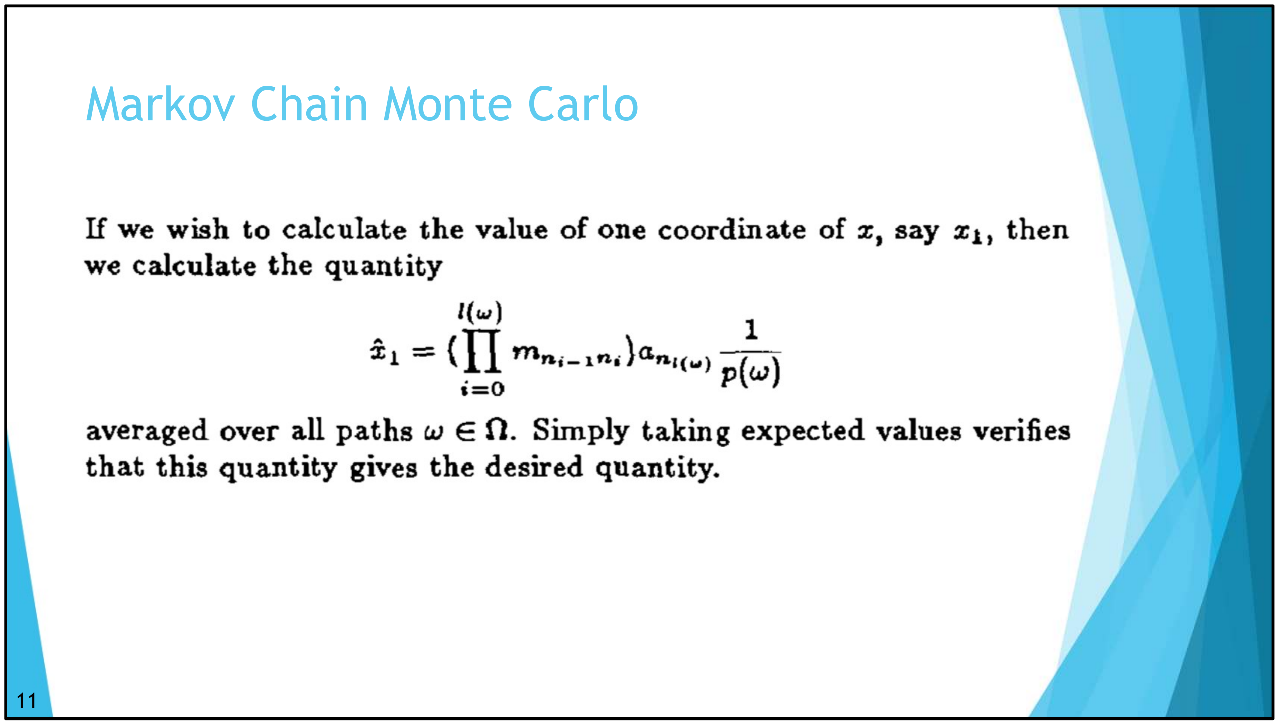
Rho is usually written *f* today—it is the BSDF

Epsilon is the emittance function, with units of W/m^2

I has units of W/m^4 .

Questions?

What is really neat is that…this is it! A little ray tracing and good choice for rho and you have a full renderer. And your code even really follows this structure. Let’s see how he turns it into code…


Markov Chain Monte Carlo

If we wish to calculate the value of one coordinate of x, say x_1, then we calculate the quantity

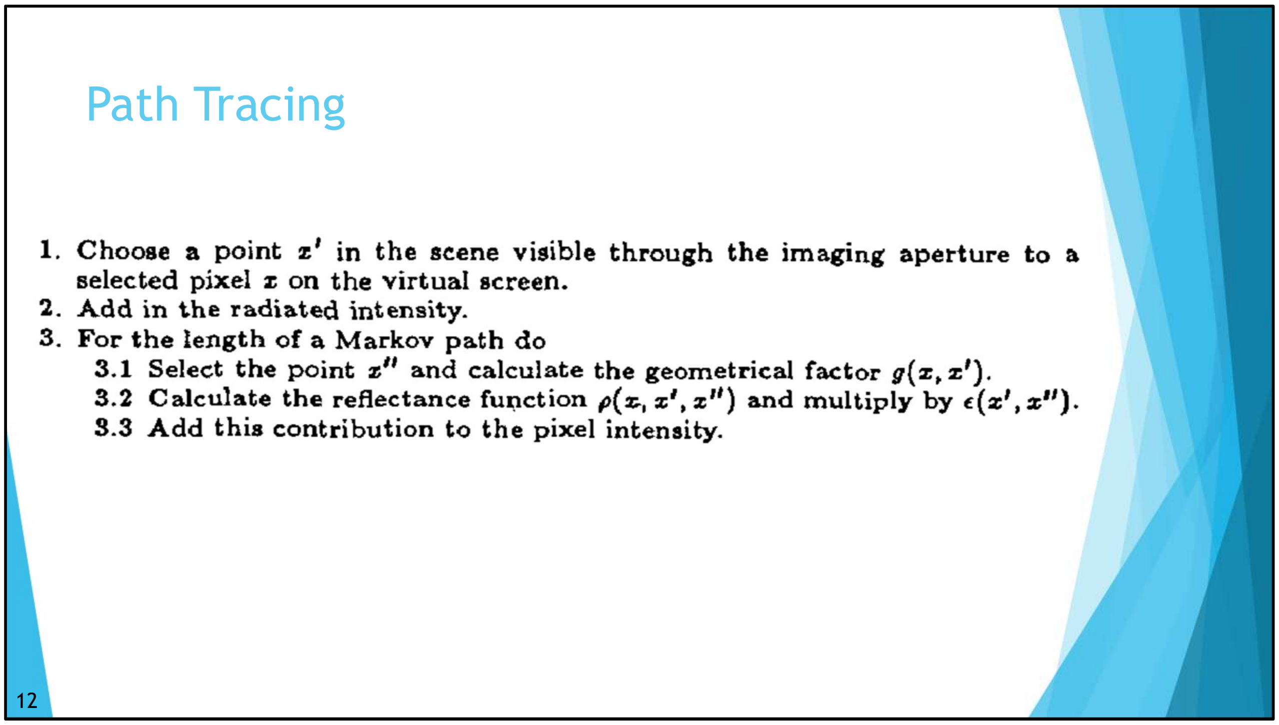
$$\hat{x}_1 = \left(\prod_{i=0}^{l(\omega)} m_{n_{i-1}n_i}\right) a_{n_{l(\omega)}} \frac{1}{p(\omega)}$$

averaged over all paths $\omega \in \Omega$. Simply taking expected values verifies that this quantity gives the desired quantity.

11

This is a weighted average of the product of the transport factors, where the weight is the inverse of the probability with which a given path was selected. Note that the denominator here is probability…this is NOT the rho variable from the previous slide.

“x” is going to be the intensity (not the point; an awkward notation choice), m is the reflectance and geometry, and a is the light emittance. This formulation assumes that light is only emitted at the end of a path, which doesn’t match the code.

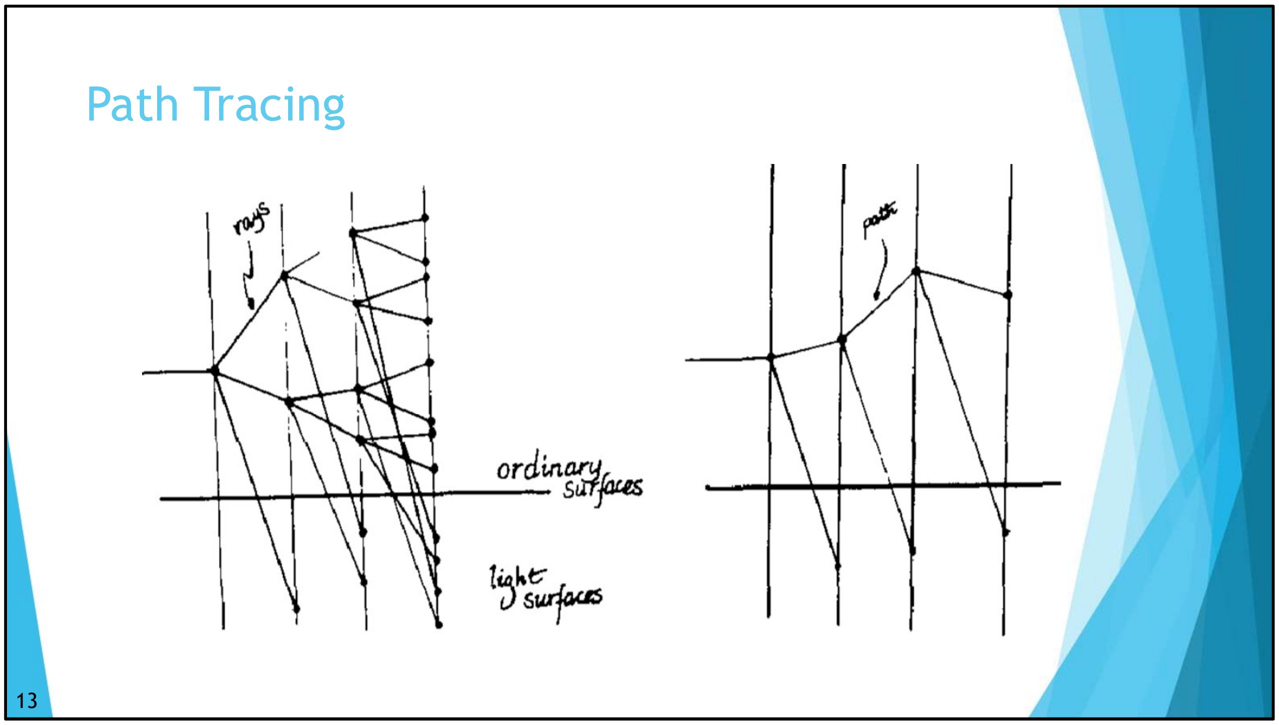


Path Tracing

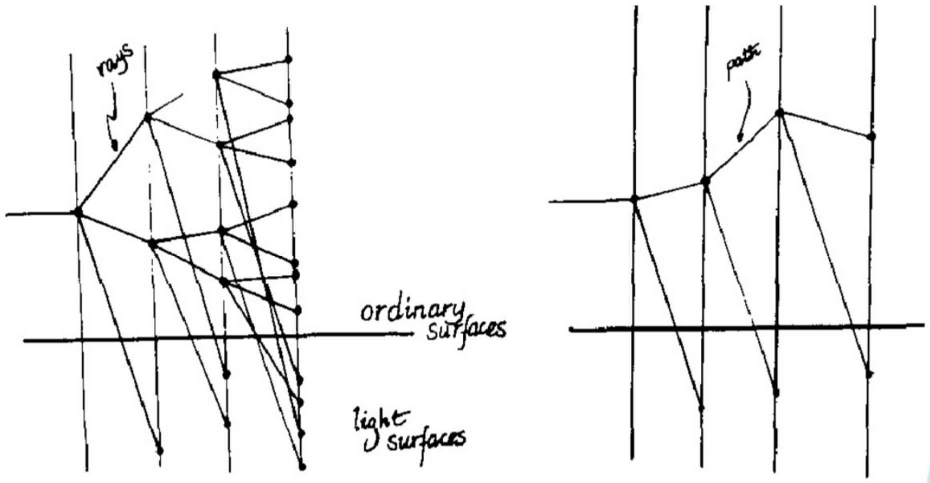
```
1. Choose a point $x'$ in the scene visible through the imaging aperture to a
   selected pixel $x$ on the virtual screen.
2. Add in the radiated intensity.
3. For the length of a Markov path do
     3.1 Select the point $x''$ and calculate the geometrical factor $g(x, x')$.
     3.2 Calculate the reflectance function $\rho(x, x', x'')$ and multiply by $\epsilon(x', x'')$.
     3.3 Add this contribution to the pixel intensity.
```

12

Here's the code. What is implicit is that the nodes on the markov path are found by ray tracing in uniformly random directions.



Path Tracing



13

This is a diagram of a search path through the scene, where the scene is the vertical axis and the iteration depth is the horizontal axis. It is a really wonderful diagram once you understand it.

You can see that previous ray tracing methods spent most of their computational time deep in the tree…which is where the incremental addition to the light intensity is *lowest* [because it is Markov]. Path tracing instead spends about equal time at all levels, and gives equal computation to direct and indirect light (as a heuristic). This is actually more in the spirit of Cook ‘84 than Cook ‘84 itself was: Kajiya is treating iteration depth as one of the dimensions of the sampling space, and ensuring that we sample whole transport paths, rather than oversampling deep and undersampling primary rays.

Results: Comparison

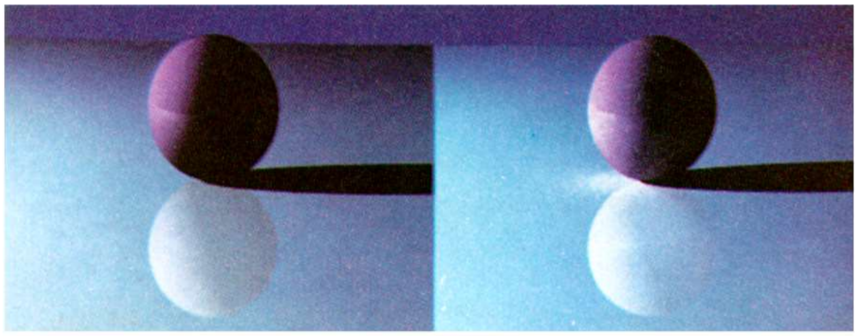


Figure 5. A comparison of ray tracing vs. integral equation technique. Note the presence of light on the base polygon scattered by the sphere from the light source.

14

Figure 5 shows a model rendered via two techniques. On the left side is the model rendered via the standard ray tracing technique (albeit with ambient coefficientsetto0~nd the singlebranching ratiospeedup mentioned above). The right image shows the result of rendering via the integral equation.

Results: New Phenomena

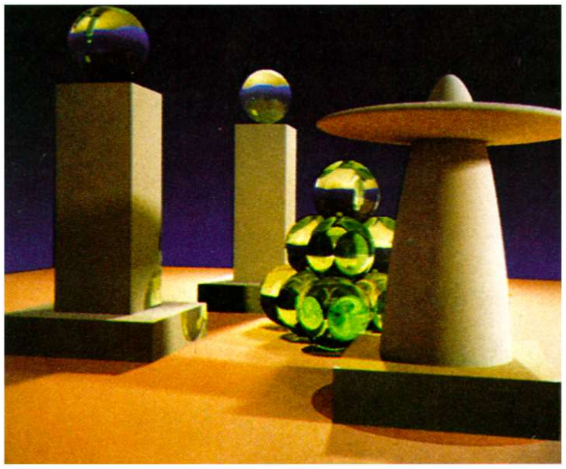
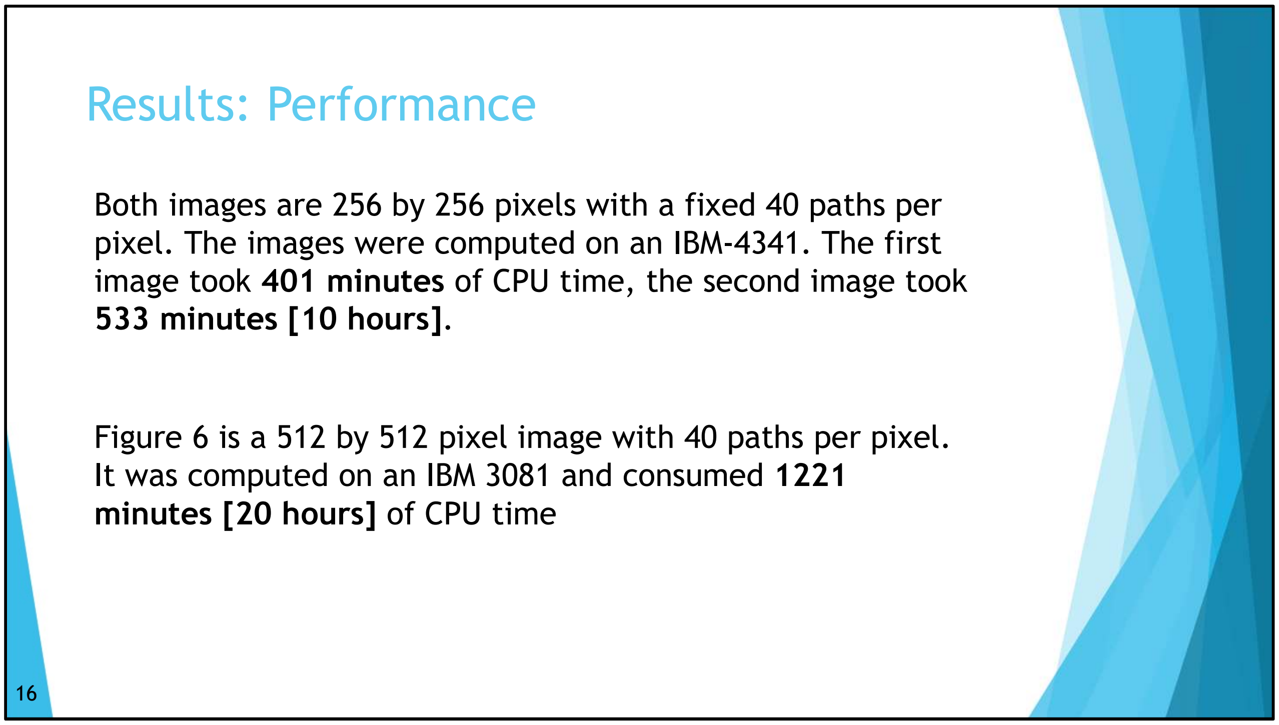


Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.

15

In figure 6 we show an image illustrating the power of the integral equation technique. All objects in the scene are a neutral grey ex- cept for the green glass bails and the base polygon (which is slightly
reddish). Any color on the grey objects would be missing from a ray tracingimage. Notethatthegreenglassballscastcausticsonobjects" in the scene. There iscolor bleeding from the lightlycolored base poly- gon onto the bottom of the oblate spheroid in the upper right. For simplicity and comparison purposes, the opaque surfaces in thisscene are lambertian, but there isno restrictionon the lightingmodels that can be used.

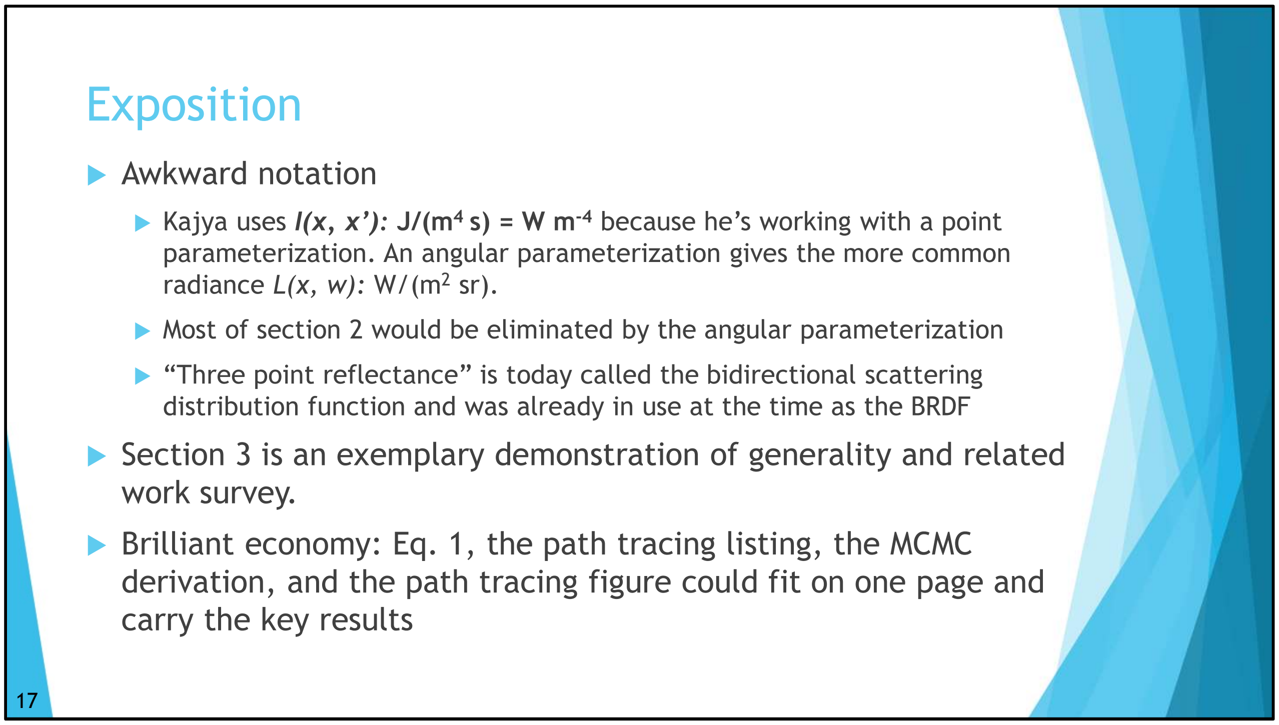


Results: Performance

Both images are 256 by 256 pixels with a fixed 40 paths per pixel. The images were computed on an IBM-4341. The first image took **401 minutes** of CPU time, the second image took **533 minutes [10 hours]**.

Figure 6 is a 512 by 512 pixel image with 40 paths per pixel. It was computed on an IBM 3081 and consumed **1221 minutes [20 hours]** of CPU time

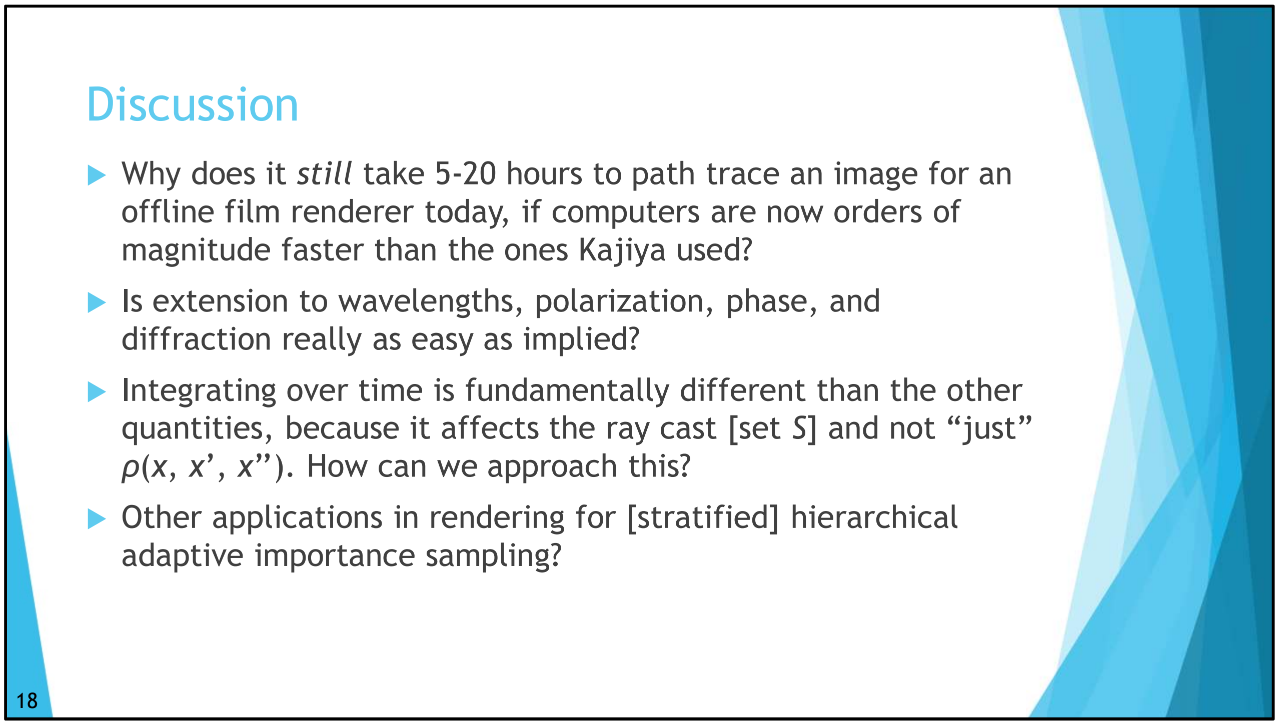
16



Exposition

- Awkward notation
  - Kajya uses ***I(x, x')*:** **J/(m^4 s) = W m^{-4}** because he's working with a point parameterization. An angular parameterization gives the more common radiance *L(x, w)*: W/(m^2 sr).
  - Most of section 2 would be eliminated by the angular parameterization
  - "Three point reflectance" is today called the bidirectional scattering distribution function and was already in use at the time as the BRDF
- Section 3 is an exemplary demonstration of generality and related work survey.
- Brilliant economy: Eq. 1, the path tracing listing, the MCMC derivation, and the path tracing figure could fit on one page and carry the key results

17



Discussion

- Why does it *still* take 5-20 hours to path trace an image for an offline film renderer today, if computers are now orders of magnitude faster than the ones Kajiya used?
- Is extension to wavelengths, polarization, phase, and diffraction really as easy as implied?
- Integrating over time is fundamentally different than the other quantities, because it affects the ray cast [set S] and not “just” $\rho(x, x', x'')$. How can we approach this?
- Other applications in rendering for [stratified] hierarchical adaptive importance sampling?

18